Was einen Compliance-gerechten Sprachassistenten von „Alexa“ unterscheidet

LOKALER BEFEHLS-EMPFÄNGER

Digitale Sprachassistenten wie Alexa, Google, Siri & Co erfreuen sich auch in Deutschland hoher Beliebtheit – Tendenz steigend. Bei allen genannten und vielen weiteren Systemen handelt es sich um cloudbasierte Architekturen – die gesprochenen Befehle werden in Rechenzentren rund um den Globus übertragen und dort interpretiert. Aus Sicht des Datenschutzes und der Privatsphäre ist das problematisch.[1] Auch die Abhängigkeit zu Cloud-Anbietern kann zu Schwierigkeiten führen, zum Beispiel wenn sich die Sprachassistenten oder Smart-Home-Geräte nicht mehr nutzen lassen, weil der Anbieter seinen Dienst einstellt. Im Rahmen eines internen Forschungsprojekts hat das Institut für Internet-Sicherheit nun einen „dezentralen“ Sprachassistenten entwickelt, der im Offline-Betrieb operiert und die Sprachdaten lokal auf dem Gerät verarbeitet, ohne sie in eine entfernte Cloud übertragen zu müssen.

Bild: ©Kaspars Grinvalds - stock.adobe.com

Sprachassistenten sind IT-Systeme, die mithilfe der natürlichen Sprachen Dienste für den Nutzer erbringen. Sie haben in den letzten Jahren immer mehr an Bedeutung gewonnen und kommen in vielen Anwendungsbereichen auf verschiedenen Endgeräten zum Einsatz. Im Allgemeinen besteht ein Sprachassistent aus verschiedenen Softwarekomponenten, welche die Ausführung von Sprachbefehlen ermöglichen. Beispielsweise können im Smart-Home-Bereich intelligente Geräte verbunden und durch Sprachbefehle gesteuert werden, im Arbeitsumfeld wird durch sprachbasierte Interaktion eine höhere Effizienz erreicht – zum Beispiel, wenn Bedienelemente nicht durch die Hände betätigt werden können. Menschen mit körperlichen Einschränkungen bringt die sprachbasierte Interaktion eine Erleichterung des Alltags.

Die Kehrseite der Medaille: Datenschutzprobleme und zum Teil auch Sicherheitsrisiken. So wurde beispielsweise ein Fall bekannt, wo sich Fremde über den Sprachassistenten Zutritt zu einem Haus verschafften.[2] Ziel des Projekts war die Entwicklung einer datenschutzfreundlichen Alternative zu cloudbasierten Sprachassistenten im Smart-Home-Umfeld, der die folgenden Anforderungen erfüllt:

- hohe Privatsphäre und Datenhoheit des Nutzers
- hohe Qualität der Spracherkennung auch ohne cloudbasierte Verarbeitung der Sprachdaten
- einwandfreie Funktionalität auch ohne Verbindung zum Internet

In den folgenden Abschnitten geht es darum, wie diese Anforderung bei der Entwicklung wdes dezentralen Sprachassistenten umgesetzt wurden.

ARCHITEKTUR DES SPRACHASSISTENTEN

Der Sprachassistent wurde mit dem Rhasspy Framework entwickelt – ein Rahmensystem, das Schnittstellen für alle notwendigen Softwarekomponenten eines Sprachassistenten zur Verfügung stellt. Dadurch wird eine einfache und schnelle Implementierung ermöglicht, ohne tiefes Grundwissen über die mathematischen Modelle von Spracherkennungsalgorithmen besitzen zu müssen.

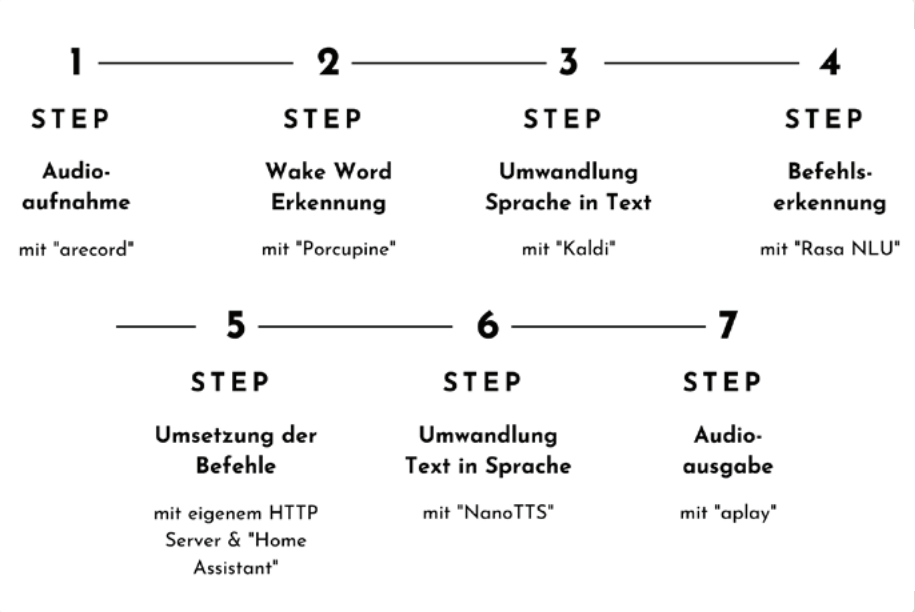

Bild 1: Workflow und Komponenten des Rhasspy-Sprachassistenten

Die Komponenten lassen sich mit Rhasspy über „Profile" konfigurieren. Ein Profil setzt sich aus den einzelnen Konfigurationen der Komponenten des Sprachassistenten zusammen. Die Komponenten spiegeln gleichzeitig den Workflow des Sprachassistenten wider (siehe Bild 1).

Die sieben Schritte umfassen folgende Funktionen:

1) **Audioaufnahme:** Aufnahme des Gesprochenen mit dem Linux-basierten Audioaufnahmetool „arecord" über ein Mikrofon und Speicherung in einer WAV-Datei
2) **Wake Word Erkennung:** Erkennung des Rufwort oder auch „Wake-Word" – zum Beispiel „Alexa", „OK Google" oder „Hi Siri" anhand der WAV-Datei
3) **Spracherkennung beziehungsweise Umwandlung von Sprache in Text:** Erzeugung einer textbasierten Darstellung der WAV-Datei mit dem Open Source Spracherkennungs-Toolkit „Kaldi"
4) **Befehlserkennung:** Erkennung des Befehls aus der textbasierten Darstellung mit „Rasa NLU", ein auf Machine Learning basiertes Toolkit für „Natural Language Understanding" (NLU)
5) **Umsetzung der Befehle:** Umsetzung der Befehle mit entsprechenden angebundenen Softwarekomponenten, zum Beispiel „Home Assistant" für die Smart-Home-Funktionalitäten
6) **Umwandlung von Text in Sprache:** Wenn eine Antwort des Sprachassistenten erforderlich ist, wird der erzeugte Text, zum Beispiel von „Home Assistant" zurück an das Rhasspy Framework übermittelt, in Sprache umgewandelt und
7) **Antworten:** in der Audioausgabe über die verbundenen Lautsprecher ausgegeben

Alle Softwarekomponenten und Optionen des Rhasspy Frameworks sind austauschbar. Es kann zum Beispiel pro Nutzer ein Profil eingerichtet werden, sodass eine Instanz mit der Spracherkennung basierend auf „Kaldi" in deutscher Sprache parallel zu einer Instanz basierend auf „DeepSpeech" in englischer Sprache laufen kann. So ist auch eine separate Weiterentwicklung der Sprachmodelle durch einzelne Nutzer möglich.

ARCHITEKTUR DES SMART-HOME-SYSTEMS

Die entwickelte Smart-Home-Architektur basierend auf dem dezentralen Sprachassistenten ist in Bild 2 dargestellt. Im internen Forschungsprojekt wurde der Sprachassistent auf einem Notebook realisiert. Die lokalen Systemkomponenten (im Bild orange gefärbt) können offline betrieben werden. Ein Webserver und Home-Assistant-Server stellen Schnittstellen nach außen, also zu internetbasierten Diensten, bereit.

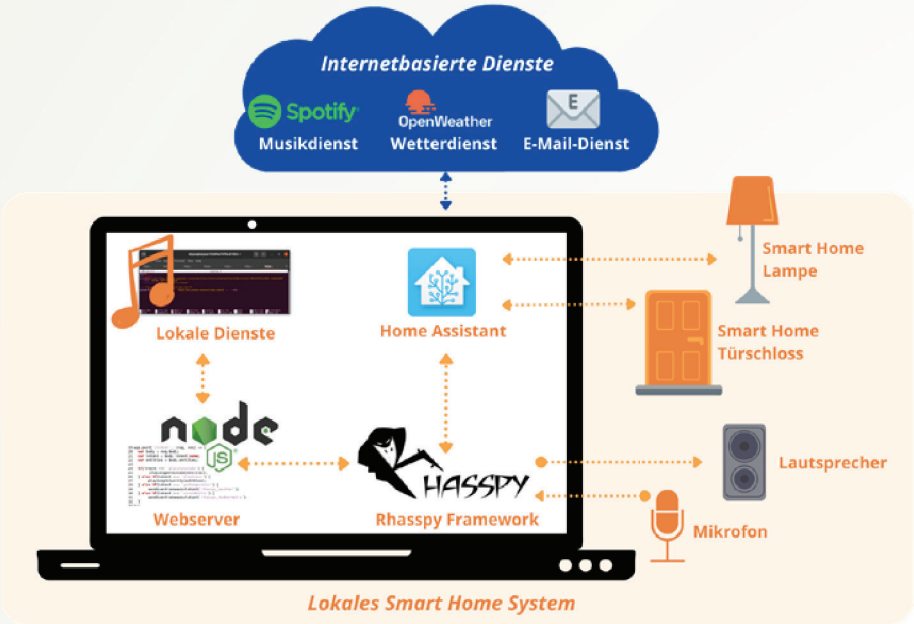

Bild 2: Smart-Home-Architektur mit dem dezentralen Sprachassistenten

Die Sprachbefehle werden von einem Mikrofon aufgenommen und mit dem Rhasspy Framework verarbeitet, wo unter anderem die Komponenten zur Sprach- und Befehlserkennung integriert sind. Die erkannten Befehle werden dann an einen Webserver weitergeleitet, welcher sich um die Weiterleitung an die entsprechenden Dienste kümmert, sodass die Befehle ausgeführt werden können. Ein Beispiel ist das An- oder Ausschalten einer Smart-Home-Lampe. Informationen, die von den Diensten bereitgestellt werden (zum Beispiel der Wetter- und E-Mail-Dienst), gelangen zurück zu Home Assistant, wo aus den Informationen ein Satz für die Sprachausgabe erzeugt wird. Dieser Satz wird zurück an das Rhasspy Framework gesendet, um den Text in eine Audioausgabe umzuwandeln und ihn über die verbundenen Lautsprecher auszugeben.

Dieser datenschutzfreundliche Sprachassistent kann auf einem Gerät im lokalen Netzwerk des Nutzers betrieben werden und benötigt keine Internetkonnektivität. Lediglich für internetbasierte Dienste ist eine Internetverbindung erforderlich. Hier kann der Nutzer aber selbstbestimmt entscheiden, welche Dienste er nutzen möchte und welche Daten er dadurch teilt. Diese Entwicklung hat gegenüber cloudbasierten Architekturen wie bei Alexa einige Vorteile: Die Daten und Befehle werden auf dem Gerät lokal verarbeitet und ausgeführt, sodass keine Übertragung durch das Internet zu einer Cloud notwendig ist. Die Privatsphäre und die Datenhoheit des Nutzers bleiben dadurch vollständig erhalten. Ebenso können auch Dienste, zum Beispiel für das Steuern von Smart-Home-Geräten über Home Assistant, lokal und ohne Internetkonnektivität betrieben werden. Trotzdem besitzt der Nutzer mit diesem System die Freiheit und Flexibilität, internetbasierte Dienste nutzen zu können. Durch den Einsatz von Open-Source-Frameworks bleibt der Sprachassistent unabhängig von Service-Anbietern, die möglicherweise ihre Dienste einstellen oder ihre Konditionen so ändern, dass sie für den Nutzer beispielsweise zu teuer werden.

SPRACHERKENNUNG MIT KALDI

Die Spracherkennung stellt die wichtigste Komponente des Sprachassistenten dar. Die unter dem Rhasspy Framework empfohlene Spracherkennungskomponente ist Kaldi – ein Open-Source-Toolkit für Automatic Speech Recognition (ASR), das nach Experimenten im Rahmen wissenschaftlicher Arbeiten im Vergleich die beste Spracherkennung aufweist.[3] Alternativen sind zum Beispiel die ASR-Komponenten „Pocket-Sphinx“ oder „DeepSpeech“, die Rhasspy ebenfalls unterstützt.

Im Standard-Modus von Kaldi müssen die Befehle, die vom Sprachassistenten erkannt werden sollen, in einer Liste definiert werden (in Form einer Initiierungsdatei namens „sentences.ini“). Für bestimmte Anwendungsfälle ist eine feste Definition der Sätze und Formulierung nicht möglich, zum Beispiel für freie Notizen. Der Nutzer müsste sich auf vorformulierte Notizen beschränken – für diesen Zweck nicht praktikabel. Für das notwendige Maß an Flexibilität kann Kaldi im „Open Transcription Modus“ betrieben werden, der durch ein deutlich größeres Sprachmodell eine freie Spracherkennung ermöglicht. Sätze müssen nicht mehr vorab definiert werden. So muss sich der Nutzer später auch nicht an eine spezifische Satzformulierung halten, damit der Sprachassistent die Befehle erkennen kann.

Während auf dem Raspberry Pi die Spracherkennung im Standard-Modus problemlos lief, konnte der Open Transcription Mode aufgrund mangelnder Rechenleistung dort nicht überzeugen. Nach dem Sprechen eines Befehls und dem darauffolgenden Beginn der Spracherkennung mit Open Transcription wurde der Raspberry Pi so stark ausgelastet, dass keine Interaktion mehr möglich war. Auf einem Rechner (Notebook) mit dem Intel i7-Prozessor ist die Spracherkennung im Open Transcription Modus hingegen problemlos möglich.

HOHE QUALITÄT DER SPRACHERKENNUNG

Zur Bewertung der Qualität der Spracherkennung muss diese gemessen werden können. Die allgemeine Methodik zur Evaluation der Spracherkennung ist die Berechnung der Word Error Rate (WER), also des Anteils nicht erkannter Wörter in einem Satz. Die Spracherkennung wird hierbei auf Audiodateien mit hoher phonetischer Variabilität angewendet, zum Beispiel verschiedene Stimmen, Aussprachen und Dialekte, die in einem Sprachkorpora bereitgestellt werden (zum Beispiel „TIMIT Corpus“: 2342 Sätze, 630 Sprecher, 8 Dialekte).

EVALUIERUNG DES STANDARDMODUS

Zur Evaluierung der Qualität der Spracherkennung mit Kaldi wurde die WER im Standardmodus und im Open-Transcription-Modus ermittelt und verglichen. Hierzu wurden dreizehn WAV-Dateien mit Befehlen (aus sentences.ini) aufgenommen und in Rhasspy zur Verarbeitung hochgeladen. Tabelle 1 zeigt die Ergebnisse des Standardmodus.

Nr.	Erwarteter Satz	Ergebnis	WER
1	Stelle meinen wecker auf elf uhr neunundfünfzig	stelle meinen wecker auf elf uhr neunundfünfzig	0
2	Spiele das Lied Sonne von Rammstein	spiele das lied sonne von rammstein	0
3	Ich möchte das Lied Hamma von Culcha Candela hören *Abweichung: Culcha Candela und Hamma sind nicht als Slot-Werte definiert*	ich möchte das lied sonne von kalt schalte sunshine live hören	0.56
4	Ich möchte das Lied 99 Luftballons von Nena hören *Abweichung: Nena und 99 Luftballons sind nicht als Slot-Werte definiert*	ich möchte das lied neunundneunzig lied sonne von live hören	0.33
5	Ich möchte 1 Live hören	ich möchte eins live hören	0
6	Ich möchte das Radio 1 Live hören *Abweichung: „Ich möchte <radiosender> hören" – „das Radio" ist nicht im Template der sentences.ini definiert*	ich möchte das radio eins live hören	0
7	Wie viele Mails habe ich *Abweichung: „Wie viele Mails hab ich" ist im Template der sentences.ini definiert*	wie viele mails hab ich	0
8	Wie viele neue Mails hab ich	wie viele neue mails hab ich	0
9	Wie wird das Wetter morgen	wie wird das wetter morgen	0
10	Wie wird das Wetter in 3 Tagen *Abweichung: „in 3 Tagen" ist nicht in einem Slot definiert*	wie wird das wetter im dreizehn wir	0.43
11	Wie spät ist es	wie spät ist es	0
12	Wie kalt ist es	wie kalt ist es	0
13	Schalte die Wohnzimmer Lampe ein	schalte die wohnzimmer lampe ein	0
		ERGEBNIS	**0.10**

Tabelle 1: Ergebnisse der Spracherkennung im Standardmodus

Nr.	Erwarteter Satz	Ergebnis	WER
1	stelle meinen wecker auf elf uhr neunundfünfzig	stelle meinen wecker auf elf uhr neunundfünfzig	0
2	Spiele das Lied Sonne von Rammstein	spiele das lied sonne von ramstein	0
3	*Ich möchte das Lied Hamma von Culcha Candela hören*	ich möchte das lied hammer von kahlschlag mandela hören	0.56
4	Ich möchte das Lied 99 Luftballons von Nena hören	ich möchte das lied neunundneunzig luft balance von nina hören	0.33
5	Ich möchte 1 Live hören	ich möchte das radio eins live fördern	0
6	Ich möchte das Radio 1 Live hören	ich möchte eins live hören	0
7	Wie viele Mails habe ich	wie viele mails hab ich	0
8	Wie viele neue Mails hab ich	wie viele neue mails habe ich	0
9	Wie wird das Wetter morgen	wie wird das wetter morgen	0
10	Wie wird das Wetter in 3 Tagen	wie wird das wetter in drei tagen	0.43
11	Wie spät ist es	wie spät ist es	0
12	Wie kalt ist es	wie kalt ist es	0
13	Schalte die Wohnzimmer Lampe ein	schalte die wohnzimmer lampe ein	0
		ERGEBNIS	**0.10**

Tabelle 2: Ergebnisse der Spracherkennung mit Open Transcription

Für die Befehle Nummer 2 bis 4 wurden Slots (variable Bereiche für Worte in einem Satz) für Interpreten und Musiktitel definiert, welche die Werte „Rammstein" und „Sonne" enthalten. Dies erklärt, wieso nur Befehl Nr. 2 im Gegensatz zu Nr. 3 und 4 korrekt erkannt wird. Die Interpreten „Culcha Candela" und „Nena" sind nicht als Werte in den Slots hinterlegt und werden nicht erkannt. Interessant ist, dass Rhasspy versucht, anhand bekannter Wörter aus der sentences.ini das Gesprochene zu transkribieren: Culcha Candela wird mit „kalt schalte sunshine live" transkribiert, eine Mischung aus den Sätzen „wie kalt ist es", dem Slot „radiosender" mit dem Wert „sunshine live" und „schalte <radiosender> an". Anstelle des Musiktitels „Hamma" wird der bekannte Wert „Sonne" transkribiert, ein Wert des Slots „song".

Befehle Nr. 3, 4, 6, 7 und 10 (grau markiert) enthalten minimale Abweichungen von den definierten Befehlen in der sentences.ini, die in der Tabelle in kursiv unter dem erwarteten Satz beschrieben sind. Nr. 6 und Nr. 7 werden korrekt erkannt, wobei auffällig ist, dass es sich bei diesen um Sätze mit minimal abweichenden Wörtern handelt. Nr. 3, 4 und 10 enthalten unbekannte Werte für Slots, die nicht erkannt werden.

Aus den Ergebnissen lässt sich schließen, dass Rhasspy die in der sentences.ini definierten Sätze ohne Open Transcription sehr gut erkennen kann und auch mit minimal abweichenden Wörtern umgehen kann. Bei unbekannten Wörtern wie „Hammer", „Luftballons" und „in 3 Tagen" aus den Befehlen Nr. 3, 4 und 10 wird das Unbekannte versucht, mit bekannten Wörtern aus der sentences.ini und den Werten der Slots zu transkribieren, was zu einer hohen WER führt.

EVALUIERUNG DES OPEN-TRANSCRIPTION-MODUS

Die Ergebnisse der freien Spracherkennung im Open-Transcription-Modus sind in Tabelle 2 dargestellt.

Die Ergebnisse zeigen, dass die Spracherkennung mit Open Transcription basierend auf den gesprochenen Testbefehlen eine bessere Spracherkennung mit einer Gesamt-WER von nur 3 Prozent liefert. Nur Befehl Nr. 3 und 4 weisen eine WER von 22 Prozent auf, die jedoch im Vergleich zu den Ergebnissen des Standard-Modus (56 Prozent und 33 Prozent) deutlich niedriger ist.

Des Weiteren ähnelt der Klang des Ergebnisses von Befehl Nr. 3 im Open-Transcription-Modus („ich möchte das lied hammer von kahlschlag mandela hören") dem erwarteten Satz („Ich möchte das Lied Hamma von Culcha Candela hören") mehr, als das Ergebnis des Standardmodus („ich möchte das lied sonne von kalt schalte sunshine live hören"). Auch die WER zeigt im Open-Transcription-Modus eine deutliche Verbesserung, was anhand der Grafik zum Befehl Nr. 4 sichtbar wird (Bild 3).

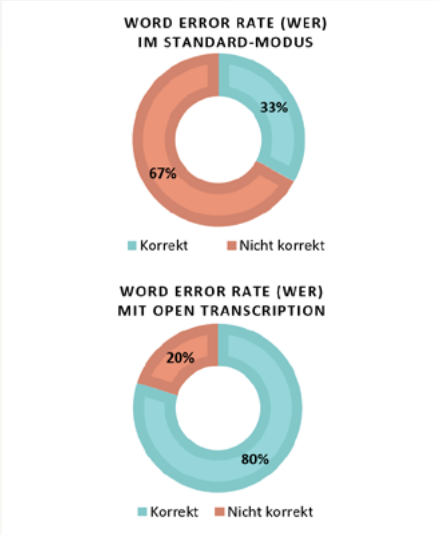

Bild 3: Verbesserung der Qualität der Spracherkennung durch Open Transcription

Zusammenfassend zeigt die Spracherkennung mit Open Transcription sehr gute Ergebnisse und ermöglicht durch die Erkennung von unbekannten Worten eine deutlich höhere Flexibilität, als im Standardmodus. Hierdurch wird die Anforderung (3) einer einwandfreien Funktionalität zum Teil erfüllt. Durch die Evaluierung konnte herausgefunden werden, dass der Betrieb von Kaldi im Open Transcription bessere Ergebnisse liefert und weniger Aufwand produziert als der Standardmodus.

Zur Erkennung englischer Musiktitel und Kunstwörter (zum Beispiel „Culcha Candela") muss das Sprachmodell von Rhasspy erweitert werden. Für eine alltägliche deutsche Sprache reicht die Spracherkennung jedoch aus. Zur weiteren Optimierung der Open-Transcription-Spracherkennung können folgende Maßnahmen vorgenommen werden:

- Konfiguration der Kaldi Parameter
- Sprachmodell verbessern (zum Beispiel durch Anpassung des Aussprache-Wörterbuchs)
- Audioaufnahme optimieren (zum Beispiel durch ein besseres Mikrofon)

EINWANDFREIE FUNKTIONALITÄT

Für eine einwandfreie Funktionalität muss der Sprachassistent die Befehle problemlos ausführen können. Das bedeutet, der lokale Sprachassistent muss in der Lage sein, die Befehle fehlerfrei zu erkennen und umzusetzen. Der Aufwand zur Gewährleistung einer einwandfreien Funktionalität sollte verhältnismäßig sein.

Die Erkennung der Befehle hängt von der Genauigkeit der Spracherkennungs-Komponente und der verwendeten Software ab. In Rhasspy kann die Software „Fsticuffs" zur Befehlserkennung verwendet werden, welche die Bibliothek rhasspy-nlu nutzt und nur die Befehle erkennen kann, auf die Rhasspy mit der sentences.ini trainiert wurde. Eine Alternative hierfür ist „RasaNLU", ein Tool, das auf Machine Learning basiert und anhand von trainierten Mustern Befehle erkennen kann. Im Gegensatz zu Fsticuffs müssen für RasaNLU ein eigener Server aufgesetzt sowie Muster definiert und trainiert werden. Dadurch steigt zwar der Aufwand, aber auch die Befehlserkennung erlangt ein größeres Maß an Freiheit, die Sätze und Slots müssen also nicht fest definiert sein, da durch die Mustererkennung auch aus unbekannten Sätzen Befehle und Slots extrahiert werden können.

Tabelle 3 zeigt die Ergebnisse der Befehlserkennung mit Fsticuffs, basierend auf der Spracherkennung im Standardmodus. Insgesamt wurden in 9 von 13 Fällen die Befehle erkannt. Die Befehle und Slots ohne Abweichungen wurden in 8 von 8 Fällen erkannt. In den 5 Fällen mit minimalen Abweichungen von den definierten Sätzen kam es jedoch zu Problemen bei der

Nr.	Umgewandelter Text	Befehl erkannt	Erkannte Slots
1	stelle meinen wecker auf elf uhr neunundfünfzig	**Ja**	**2/2 (**elf, neunundfünzig)
2	Spiele das Lied Sonne von Rammstein	**Ja**	**2/2** (Sonne, Rammstein)
3	Ich möchte das Lied Hamma von Culcha Candela hören	**Nein**	**0/2** (Hamma, Culcha Candela)
4	Ich möchte das Lied 99 Luftballons von Nena hören	**Nein**	**0/2** (99 Luftballons, Nena)
5	Ich möchte 1 Live hören	**Ja**	**1/1 (**1 Live)
6	Ich möchte das Radio 1 Live hören	**Ja**	**0/1** (1 Live) *„das Radio" statt "1 Live" wurde als Slot „radiosender" erkannt, was auf die Satzstellung zurückzuführen ist, in dem Template folgt nach „ich möchte" der Radiosender*
7	Wie viele Mails habe ich	**Nein**	Keine *Abweichung: „hab" statt „habe" definiert*
8	Wie viele neue Mails hab ich	**Ja**	Keine
9	Wie wird das Wetter morgen	**Ja**	**1/1** (morgen)
10	Wie wird das Wetter in 3 Tagen	**Nein**	**0/1** (in 3 Tagen)
11	Wie spät ist es	**Ja**	Keine
12	Wie kalt ist es	**Ja**	Keine
13	Schalte die Wohnzimmer Lampe ein	**Ja**	**2/2** (Wohnzimmer Lampe, ein)
	ERGEBNIS	**9/13**	**8/13**

Tabelle 3: Ergebnisse der Befehlserkennung mit Fsticuffs

Erkennung: In nur einem Fall konnte der Befehl erkannt werden, aus den Slots hingegen wurde kein einziger Wert erkannt. Bei Befehl Nr. 6 wurde der Ergebnisslot mit dem falschen Wert gefüllt, was sich auf die Satzstellung in der sentences.ini zurückführen lässt.

Tabelle 4 zeigt die Ergebnisse der Testversuche mit RasaNLU mit den gleichen Befehlen, die für Fsticuffs verwendet wurden. Die Befehle wurden mit RasaNLU in 13 von 13 Fällen korrekt erkannt und auch die Slots wurden immer richtig befüllt.

Die Interpreten „Culcha Candela" und „Nena" wurden zwar nicht korrekt transkribiert, im Gegensatz zu Fsticuffs konnte RasaNLU aber genau die Wörter, welche die Interpreten bestimmen, dem entsprechenden Slot zuordnen (siehe Befehl Nr. 3 und 4).

Auch die Befehle Nr. 6 und 7 konnte RasaNLU im Gegensatz zu Fsticuffs korrekt erkennen und „1 Live" dem Slot „radiosender" zuordnen. Befehl Nr. 1, 9 und 10 zeigen die Vorteile von „Pipelines", eine erweiterte Funktionalität von RasaNLU: Die „Duckling Pipeline" ermöglicht die Extrahierung von Daten, Zeiten und Zeitdauern. In dem Satz „Regnet es morgen" erkennt diese beispielsweise das Stichwort „morgen" und stellt über den Slot „day" das entsprechende Datum in Form eines Zeitstempels bereit, die zum Beispiel für Anfragen gegen eine Wetter-API (Programmierschnittstelle) verwendet werden können.

Zusammenfassend lässt sich durch die Evaluierung sagen, dass die Befehlserkennung mit RasaNLU zusammen mit der Spracherkennung im Open Transcription Mode sehr gute Ergebnisse liefert und die Anforderung einer hohen Qualität und einwandfreien Funktionalität auch ohne Internetkonnektivität und cloudbasierte Spracherkennung erfüllt. Für die Umsetzung der Befehle müssen entsprechende Softwarekomponenten an den Sprachassistenten angebunden werden. Die Integration von Smart-Home-Funktionalitäten wurde mit dem „Home Assistant"-Framework realisiert, welches Schnittstellen für viele Smart-Home-Geräte und -Dienste zur Verfügung stellt. Zudem erkennt Home Assistant Sensoren und Kommunikationsprotokolle im Umfeld und bindet sie automatisch ein. Des Weiteren lassen sich über Rhasspy auch beliebige andere Dienste an den Sprachassistenten anbinden. Dies wurde in dem internen Forschungsprojekt anhand eines Wetter-, E-Mail- und Musikstreaming-Dienst gezeigt, auf die im nachfolgenden Abschnitt genauer eingegangen wird.

Nr.	Umgewandelter Text	Befehl erkannt	Erkannte Slots
1	stelle meinen wecker auf elf uhr neunundfünfzig	Ja	**1/1** *Richtiger Timestamp wurde extrahiert mit der Duckling Pipeline*
2	Spiele das Lied Sonne von Rammstein	Ja	**2/2** (Sonne, Rammstein)
3	Ich möchte das Lied Hamma von Culcha Candela hören	Ja	**2/2** (Hamma, Culcha Candela) *Es wurde „hammer" in den Slot „song" und „kahlschlag mandela" in den Slot „interpret" gefüllt*
4	Ich möchte das Lied 99 Luftballons von Nena hören	Ja	**2/2** (99 Luftballons, Nena) *Es wurde „neunundneunzig luft balance" in den Slot „song" und „nina" in den Slot „interpret" gefüllt*
5	Ich möchte 1 Live hören	Ja	**1/1 (1 Live)**
6	Ich möchte das Radio 1 Live hören	Ja	**1/1 (1 Live)**
7	Wie viele Mails habe ich	Ja	*Keine*
8	Wie viele neue Mails hab ich	Ja	Keine
9	Wie wird das Wetter morgen	Ja	**1/1** (morgen) *Richtiger Timestamp wurde extrahiert mit der Duckling Pipeline*
10	Wie wird das Wetter in 3 Tagen	Ja	**1/1** (in 3 Tagen) *Richtiger Timestamp wurde extrahiert mit der Duckling Pipeline*
11	Wie spät ist es	Ja	Keine
12	Wie kalt ist es	Ja	Keine
13	Schalte die Wohnzimmer Lampe ein	Ja	**2/2** (Wohnzimmer Lampe, ein)
	ERGEBNIS	**13/13**	**13/13**

Tabelle 4: Ergebnisse der Befehlserkennung mit RasaNLU

HOHE PRIVATSPHÄRE UND DATENSCHUTZ

Der dezentrale Sprachassistent schützt die Privatsphäre, da er keine Daten zur Spracherkennung an einen externen Server oder in eine Cloud sendet. Daher wird auch die Spracherkennung auf dem Gerät lokal durchgeführt. Die Integration von Smart-Home-Geräten benötigt ebenfalls keine Cloudanbindung – sie funktioniert im lokalen Netzwerk des Betreibers. Kommunikation findet nur innerhalb des Netzwerks im Home statt.

Weiterhin ist für eine hohe Privatsphäre wichtig, nur Smart-Home-Geräte zu integrieren, die ohne Internetkonnektivität betrieben werden können.[4] Die Nutzung internetbasierter Dienste ist somit die einzige Schnittstelle, über die Daten nach außen gelangen können. Hierzu müssen die Dienste vor der Anbindung hinsichtlich der Datennutzung analysiert und verglichen werden, um die datenschutzfreundlichste Alternative zu wählen. Des Weiteren müssen Dritte aus der Anbindung ausgeschlossen werden, was besonders bei der zentralen Authentifizierung mithilfe von ID-Providern eine wichtige Rolle spielt. Um dem Nutzer einen schnellen Zugriff auf einen Dienst mit möglichst geringem Aufwand zu ermöglichen, wird häufig eine Anmeldung über ID-Provider wie Google oder Facebook angeboten.

Um das hohe Maß an Privatsphäre des lokalen Sprachassistenten aufrecht zu erhalten, sollte diese Möglichkeit jedoch vermieden werden (Bild 4). Drittanbieter können „mitlesen" und

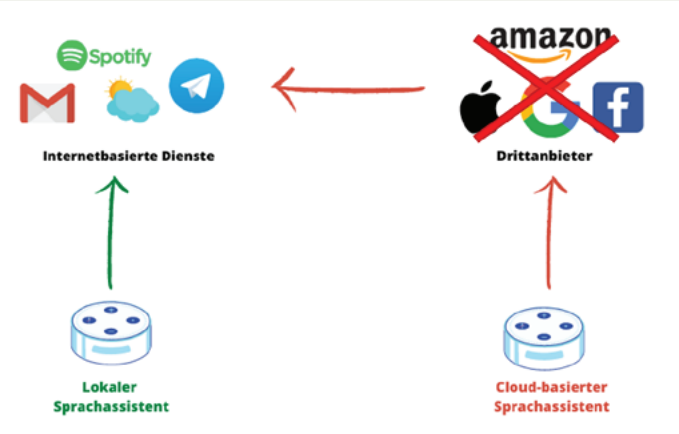

Bild 4: Datenfluss bei lokalen und cloudbasierten Sprachassistenten

erhalten durch den Authentifizierungsdienst Informationen über die vom Nutzer verwendeten Dienste und damit auf die Interessen des Nutzers. Außerdem wird der Drittanbieter durch jede Authentifizierung informiert, wann und wie oft der Nutzer einen Service nutzt. Möglicherweise erhält dieser sogar das Recht, Anfragen mitzulesen und Details über den Nutzer und sein Verhalten zu erlangen. Aus diesem Grund ist es wichtig, dass für eine hohe Privatsphäre keine Drittanbieter-Authentifizierung genutzt wird und für jeden Dienst ein eigener Account verwendet wird. So werden die von den Diensten gesammelten Daten abgekapselt und nur für den jeweiligen Zweck des Dienstes verwendet.

Um zu zeigen, wie internetbasierte Dienste mit einer hohen Privatsphäre angebunden werden können, wurden im Forschungsprojekt drei internetbasierte Dienste an den dezentralen Sprachassistenten angebunden:

- Datenschutzfreundlicher Dienst: Wetterdienst ohne Nutzerauthentifizierung und -tracking,
- Datenschutzfreundliche Anbindung: Anbindung eines E-Mail-Dienstes über das IMAP-Protokoll, statt Mail-APIs (zum Beispiel Google Mail), um keine Dritten in die Mailabfrage zu involvieren.
- Datenschutzfreundliche Implementierung: Implementierung eines datenschutzfreundlichen Musikstreaming-Dienstes mit „YouTube-dl", ein Open Source Projekt, das ohne Nutzerauthentifizierung und -tracking Videos anhand der URL abspielen kann und über die Kommandozeile durch den Sprachassistenten aufgerufen wird.

Eine eigene lokale Implementierung der Funktionen bietet sich an, insofern sich der Dienst ohne großen Verlust an Funktionalität nachbauen lässt. Anhand der eigenen Implementierung des Musikstreamings lassen sich die Differenzen im Vergleich zu internetbasierten Diensten wie Spotify leicht erklären. Spotify kann durch die Speicherung der Daten des Nutzers einen komfortableren Dienst mit vielen verschiedenen Funktionalitäten ermöglichen, zum Beispiel das Favorisieren von Musiktiteln, die Erstellung von Playlists und Musikvorschläge. Hingegen kann der lokale Dienst lediglich einen Musiktitel suchen und abspielen. Der Aufwand, eine gleichwertige lokale Alternative zu implementieren, ist sehr hoch, weshalb hier zwischen dem Maß an Funktionalität und dem gewünschten Datenschutzniveau abgewogen werden muss.

FAZIT

Der entwickelte dezentrale Sprachassistent ermöglicht eine hohe Privatsphäre des Nutzers, eine einwandfreie Funktionalität und eine hohe Qualität der Spracherkennung.

Um die Qualität der Spracherkennung mit Open Transcription zu verbessern, kann das zugrunde liegende Sprachmodell erweitert und angepasst werden. Für kleinere Anwendungsfälle, bei denen eine feste Definition der Formulierungen von Befehlen ausreicht, eignet sich auch die Spracherkennung im Standardmodus.

Um die Funktionalität des zu erweitern, müssen in einer weiterführenden Arbeit Anwendungsfälle untersucht werden. Es gilt herauszufinden, welche Dienste und Geräte die Nutzer in einzelnen Anwendungsgebieten benötigen. Der Aufwand zur Implementierung ist abhängig von den jeweiligen Diensten. Da der Prototyp aber bereits als Grundgerüst verwendet werden kann, definiert sich der Aufwand lediglich durch die Zeit zur Implementierung oder datenschutzfreundlichen Anbindung der Dienste an den lokalen Sprachassistenten. ■

Literatur

[1] *J.-H. Frintrop, N. Pohlmann: „Alexa, wie sicher bist du? – Intelligente Sprachsteuerung unter der Lupe", IT-Sicherheit – Fachmagazin für Informationssicherheit und Compliance, DATAKONTEXT-Fachverlag, 6/2017*

[2] *N. Pohlmann: „Cybersicherheit - Das Lehrbuch für Konzepte, Mechanismen, Architekturen und Eigenschaften von Cybersicherheitssystemen in der Digitalisierung" 2. Auflage, Springer Vieweg Verlag, Wiesbaden 2022*

[3] *D. Gaida, P. Lange, R. Petrick, P. Proba, A. Malatawy, D. Suendermann-Oeft: „Comparing Open-Source Speech Recognition Toolkits", 11th International Workshop on Natural Language Processing and Cognitive Science, 2014*

[4] *N. Pohlmann: „Chancen und Risiken von Smart Home", DuD Datenschutz und Datensicherheit – Recht und Sicherheit in Informationsverarbeitung und Kommunikation, Vieweg Verlag, 2/2021*

TATJANA HÜSCH
studiert im Master Internet-Sicherheit an der Westfälischen Hochschule Gelsenkirchen und beschäftigt sich im Rahmen des Studiums mit dezentralen Sprachassistenten.

NORBERT POHLMANN,
Informatikprofessor für Cybersicherheit und Leiter des Instituts für Internet-Sicherheit – if(is) an der Westfälischen Hochschule in Gelsenkirchen sowie Vorstandsvorsitzender des Bundesverbands IT-Sicherheit – TeleTrusT und im Vorstand des Internetverbandes – eco.